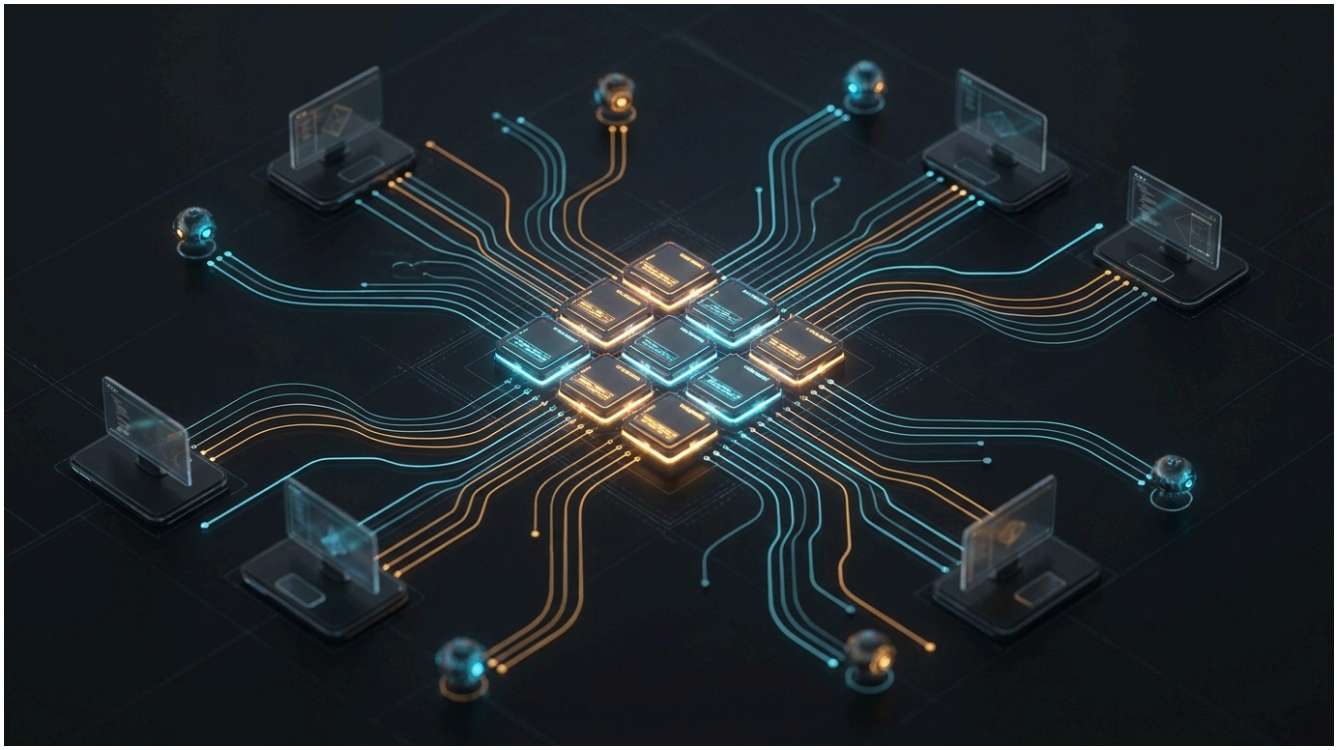
CASE STUDY · PROOF OF PLAY

A shared operating system of AI skills my whole team plugged into.

I built a Claude Code skill marketplace so every engineer could run sophisticated AI workflows, the full build-test-review-ship loop, live business reports, and balance simulations, without reinventing them or blowing their context window.

Role: Special Projects Lead | Internal platform | 4 plugin suites, ~30 skills | Adopted team-wide



One shared marketplace of agentic skills; each engineer installs only the ones they need.

The problem

AI leverage was siloed. The most useful workflows, a rigorous build-and-review loop, reports that read live business data, a combat balance simulator, lived in one person's head or one person's prompt history. Everyone else either reinvented them, used a weaker version, or pasted so much context into a single session that the model lost the thread. The result was inconsistent quality and a ceiling on how much the team could actually get out of AI.

The fix could not be “one giant mega-prompt.” It had to be **modular**: a shared library of focused skills that any engineer could install on demand, run the same way every time, and compose, without each one carrying the weight of all the others.

What I built

A **Claude Code plugin marketplace** for the company: a curated set of agentic skills, organized into suites, that an engineer installs from a single source. You pull in only what your task needs, so your context window stays clean and the workflow runs identically for everyone. Four suites cover the work:

Dev loop

The everyday engineering loop as one command. `/dev` runs the full arc, clarify the task, plan it, critique the plan, implement, test, review, open the PR, and triage feedback, with focused skills underneath it: `review-local`, `review-pr`, `review-triage`, `scrutinize`, `adversary`, `draft-pr-ticket`, `sync-base`, `fix-line`. One engineer plus this loop ships like a small team.

Operational reports, wired to live data

Reports that used to be manual analyst work, now one command each, reading straight from the systems of record: crash reports (Sentry + Amplitude), desync reports, daily summaries, push-notification performance (Braze + Amplitude), app-store rating reports (stores + Sentry + Linear), paid-marketing and growth reports, guild-chat analysis, and API-latency, with an `allreports` orchestrator that runs the whole briefing.

Game-design tooling

Designer-facing skills built on the simulation work: `balancelab` (a what-if combat simulator), `balancelab-benchmark` (replays real player runs against new content), and unlocks documentation and auditing. Designers ask balance questions without pulling an engineer.

Comms

Audience-reframing skills like `ceoify` that turn dense engineering output into the brief a given reader actually needs.

How it works

The architecture is the point. A single marketplace holds the skills; each engineer installs only the suite or skill they need; the heavy context lives inside the skill, not in the person's session. New capability ships once and the whole team has it.

One marketplace, four suites, installed on demand. Capability ships once; the whole team gets it, without anyone overloading a single session.

The `/dev` loop: a single command walks a task from clarify to triaged PR, with critic and review gates built in.

The value it added

- **AI leverage stopped being siloed.** A capability one person figured out became something the whole team could run, the same way, the same day.
- **Faster loops, fewer regressions.** The built-in critic and review gates caught problems before they reached a human reviewer, so the loop moved faster *and* safer.
- **Reports became commands.** Briefings that used to be manual analyst work now read live data on demand, so the team spent attention on decisions, not on gathering.
- **Designers got self-serve tooling.** Balance questions got answered by simulation instead of by pulling an engineer off feature work.
- **Clean context windows.** Install-only-what-you-need meant the model stayed sharp instead of drowning in everything at once.

Why this matters beyond one team

The artifact is a skill marketplace; the capability is bigger. Most organizations adopting AI end up with a few power users and everyone else left behind, because the good workflows never get packaged and shared. What I built is the opposite: a **shared AI operating model** where leverage is captured once, distributed to everyone, and improved centrally. Standing that up, deciding what to package, designing the install-on-demand model, and getting a team to actually run on it, is a repeatable capability, not a one-off script. It is exactly the kind of system I would set up for a portfolio company that wants its whole team operating at the AI frontier, not just its loudest engineer.

By the numbers

~30
Agentic skills across four suites

4
Suites: dev loop, reports, game-design, comms

1
Command (`/dev`) for the full clarify-to-PR loop

7+
Live data sources wired into reports (Sentry, Amplitude, Braze, Linear, app stores, and more)

Team-wide
Adopted across the engineering org, not a personal tool

Install
Only what you need, so context windows stay clean

> *Built at Proof of Play and run by the engineering team. The same pattern, capture AI leverage once and distribute it to everyone, is one of the highest-leverage things I can stand up for an organization. Happy to walk through the architecture in a conversation.*

Ben Young · Special Projects Lead, Proof of Play

benyoung.ai · **o.benjamin.young@gmail.com**